Breast Cancer Detection

Minor project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

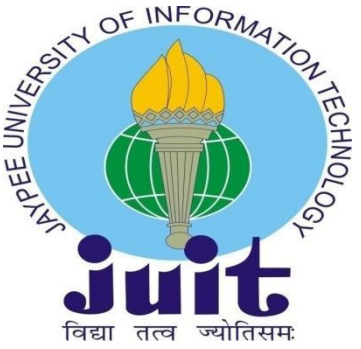
Computer Science and Engineering

By

PARUL SHARMA(191206)
KUNIKA SHARMA(191227)
RIA MAHAJAN(191236)

UNDER THE SUPERVISION OF

RAVINDARA BHATT



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, 173234, Himachal Pradesh, INDIA

TABLE OF CONTENT

DECLARATION

I hereby declare that this project has been done by me under the supervision of Dr. Ravindara Bhatt**, Affiliation,** Jaypee University of Information Technology. I also declare that neither this project nor any part of this project has been submitted elsewhere for the award of any degree or diploma.

Supervised by:

(Dr. Ravindara Bhatt)

Associate Professor

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology

Submitted by:

Parul Sharma(191206)

Kunika Sharma(191227)

Ria Mahajan(191236)

Computer Science & Engineering Department

Jaypee University of Information Technology

CERTIFICATE

This is to certify that the work which is being presented in the project report titled “Breast Cancer Detection” in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by “Parul Sharma(191206), Kunika Sharma(191227), Ria Mahajan(191236)” during the period from January 2022 to May 2022 under the supervision of Dr. Ravindara Bhatt, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Parul Sharma(191206)
Kunika Sharma(191227)
Ria Mahajan(191236)

The above statement made is correct to the best of my knowledge.

(Dr. Ravindara Bhatt)
Associate Professor
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat,

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to Almighty God for His divine blessing makes us possible to complete the project work successfully.

We are really grateful and wish our profound indebtedness to Supervisor **Dr. Ravindara Bhatt, Associate Professor**, Department of CSE Jaypee University of Information Technology, Waknaghat. The deep Knowledge & keen interest of our supervisor in the field of "**Machine Learning**" encouraged us to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, and reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

We would like to express our heartiest gratitude to **Dr. Ravindara Bhatt,** Department of CSE, for his kind help to finish my project.

We would also generously welcome each one of those individuals who have helped straightforwardly or in a roundabout way in making this project a win. In this unique situation, we might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated our undertaking.

Finally, we must acknowledge with due respect the constant support and patients of my parents.

Parul Sharma(191206)
Kunika Sharma(191227)
Ria Mahajan(191236)

ABSTRACT

Breast cancer is the world's second leading cause of cancer deaths. It begins when cells in the breast begin to proliferate uncontrollably. These cells typically form tumours, which can be seen on an X-ray or felt as lumps in the breast. Early diagnosis and detection, on the other hand, can increase the likelihood of successful treatment and survival. The key challenge to detection is determining whether tumours are malignant (cancerous) or benign (non-cancerous). A tumour is considered malignant if the cells can spread to other parts of the body or grow into them. In contrast to cancerous tumours, benign tumours do not invade nearby tissue or spread to other parts of the body. However, benign tumours can be dangerous if they press on vital structures like blood vessels or nerves. Despite the fact that successful detection of malignant tumours from histopathological pictures is mostly dependent on radiologists' long-term knowledge, specialists occasionally disagree with their decisions. A computer-aided diagnosis is a second option for image diagnosis that can help experts make more reliable decisions. In clinical applications for identifying malignant tumours from histopathological images, automatic and precision classification for breast cancer histological images are critical. Advanced convolution neural network technology has been widely employed in biomedical image processing and has had considerable success in natural image classification. In this work, we propose an accurate and inclusive computational breast cancer diagnosis by comparing the accuracies of different machine learning and deep learning models on structured data and histopathological microscopy images respectively. The machine learning models like KNN and SVM achieve an accuracy of 96.27% and 93.7% respectively on the Breast Cancer Wisconsin dataset. Furthermore, three deep learning models are proposed and analysed. The simulation results showed that our first custom-based CNN model achieves comparatively lesser accuracy (nearly 70%). The other two models employ transfer learning techniques of the powerful ResNet-50 and VGG-16 Convolutional Neural Networks, pre-trained on ImageNet to train and classify the BreakHis dataset(Histopathological Image Dataset) into benign or malignant. The resultant accuracies were obtained to be 92.17% and 97.96% respectively.

Chapter 01: INTRODUCTION

1.1 Introduction

Breast cancer is the most prevalent cancer diagnosed in women, accounting for 30% of all new cancer diagnoses in women (excluding skin cancers). Breast tissue samples allow doctors to examine the tissue's microscopic structure and components histologically. Hematoxylin and eosin (H&E) is the primary stain of tissue specimens for routine histopathological diagnoses. It is used to identify between normal tissue, non-malignant (benign) and malignant (carcinomas) lesions, as well as to perform a prognosis evaluation. Breast carcinomas come in a variety of shapes and sizes, each with its own tissue morphology. Breast carcinomas grow from the mammary epithelium and produce ductal carcinoma in situ, which is a premalignant epithelial proliferation inside the ducts. The ability of cancer cells to break past the basal membrane characterises invasive carcinoma.

The pathologist used to undertake morphological evaluation and tumour grading visually, but this technique is time-consuming and subjective, resulting in inter-observer variances even among experienced pathologists. Because the use of morphological criteria in visual categorization is subjective, computer-aided diagnosis (CAD) systems are used to enhance diagnostic accuracy, minimise human error, boost interobserver agreement, and improve repeatability. For digital pathology picture analysis, a variety of approaches have been developed, ranging from rule-based to machine learning applications. Deep learning-based systems have recently been demonstrated to outperform traditional machine learning methods in a variety of image analysis tasks, automating the entire process.

Convolutional neural networks (CNN) have been effectively applied in the medical imaging area for diabetic retinopathy screening, bone disease prognosis and age assessment, and other challenges. Previous deep learning-based applications in histological microscopic image processing have shown promise in assisting in the diagnosis of breast cancer. We provide a method for histology microscope image analysis for breast cancer type categorization in this research. Our method uses deep CNNs for feature extraction and gradient boosted trees for classification, and it outperforms other similar approaches to our knowledge.

In radiology and other medical science fields, machine learning-based algorithms play an important role in diagnosing disease in a much simpler way than ever before, thus providing a viable alternative to surgical biopsy for breast tumours. We attempted to identify and classify breast tumours in this research, comparing the outcomes of binary and multi-class classification of breast tumours with and without Transfer Learning, utilising pre-trained Keras models such as VGG16 and Convolutional Neural Network (CNN) architecture.

1.2 Objective

The purpose of this research is to use Histopathological pictures to identify tumours and cancers in the breasts and classify them. The primary goal of breast cancer and tumour detection is to help in clinical diagnosis. The goal is to develop an algorithm that ensures the presence of a tumour and classifies it using a combination of processes to give a reliable way of tumour identification in Histopathological breast pictures. The resulting picture will be able to reflect certain key information, allowing the personnel to make more informed decisions about the curing method.
We also tried to visualise the dataset on the Machine Learning Algorithms based on K-Nearest Neighbour (KNN) and Support Vector Machine (SVM) on the structured data of Wisconsin, USA.
Finally, the proposed model uses a Convolution Neural Network based on a custom based CNN model, ResNet-50 and VGG - 16to determine whether a particular histopathological brain picture includes a tumour or not.

1.3 Motivation

Maintaining one's health is crucial in today's world. The increased prevalence of health issues has put a considerable strain on doctors throughout the world, making it more difficult to treat each patient successfully. To reduce the strain on doctors, there have been several significant new innovations that have revolutionised the healthcare industry. Because of advancements in computer technology, more accurate and faster results may be obtained, and patients can be treated accordingly. Early identification of cancer remains a challenge in healthcare. According to the World Health Organisation, cancer cells can be detected and destroyed if they are discovered in their early stages. As a result, early detection of cancer cells is critical for reducing cancer-related problems.

Patients with breast cancer are at the very top of the cancer patient list. Because many women are at risk for breast cancer, finding and eliminating cancer cells from the patient's body is crucial. According to the World Health Organisation, breast cancer claimed the lives of over a million women in 2011, and the situation is only getting worse. Breast cancer is the second most common cancer in both women and men throughout the world. It was responsible for around 12% of all new cancer cases in 2012, and 25% of all cancers in women. Breast cancer risk rises when cells in the breast begin to grow out of control. These cells usually develop into a tumour, which can be seen on x-rays or felt as a lump on the skin.

This research aims to use and analyse a variety of Machine Learning and Deep Learning algorithms in order to develop a solution that can assist a patient in determining whether she is at risk for breast cancer at an early stage, allowing the breast cancer cells to be eliminated with appropriate medication. Neurosurgeons and healthcare experts can utilise the system. The method, which combines image processing and pattern analysis, is projected to enhance breast cancer screening performance metrics. The main objective of medical imaging projects is to extract as much useful and accurate information as possible from these pictures with the least amount of inaccuracy. The right combination and parameterization of the phases allow for the construction of a tool that can aid in tumour early detection or surveillance.

1.4 Language Used

Python 3 - The main reasons for the use of this programming language for the implementation of the project are stated below:

Simple and dependable:

Python provides code that is both concise and readable. Machine learning and AI are based on sophisticated algorithms and flexible workflows, and Python's simplicity allows developers to design dependable solutions. Instead of focusing on the technical subtleties of the language, developers can devote all of their attention to solving an ML problem.

Extensive selection of libraries and frameworks:

To enable developers to come up with the greatest coding solutions, it's critical to have a well-structured and well-tested environment

Python makes a large number of libraries and frameworks available.

These frameworks and libraries are used by programmers to reduce development time. Here are a few examples:

Machine learning frameworks include Keras, TensorFlow, and Scikit-learn.

- NumPy is a Python package for scientific computing and data analysis.
- SciPy is a Python package for advanced computation.
- Pandas is a data analysis tool that can be used for a variety of purposes.
- Seaborn is a data visualisation platform.

Independency of platform

Platform independence refers to a programming language or framework that allows developers to create things on one system and then utilise them on another with few (or no) modifications. Python's success stems from the fact that it is a platform-independent language. It is available on a variety of operating systems, including Linux, Windows, and macOS. utilizedPython code may be used to produce standalone executable programs for the majority of mainstream operating systems, allowing Python software to be distributed and utilised without the need for a Python interpreter.

1.5 Technical Requirements (Hardware)

1.5.1 Software Requirements: For the reasons described above, Python 3 was utilised to construct the project. The dataset was obtained using Kaggle. In the event of programming syntax mistakes, GitHub and StackOverflow were used as resources.

Google Colaboratory is a high-GPU Jupyter Notebook interface that is open-source. Google Colab is a free Jupyter notebook environment that runs fully in the cloud and requires no setup. Colab allows users to develop and execute code, store and share analysis, and access sophisticated computational resources all from the browser, all for free. Jupyter Notebook is a useful tool for iterating and writing Python data analysis scripts. Instead of creating and rewriting a full programme, lines of code may be written and run one at a time.

1.5.2 Hardware Requirements:

Processor: Intel® Core™ i5-10300H

Installed memory (RAM):8.00GB

System Type: 64-bit Operating System

1.6 Outcomes

Our study intends to use several machine learning (KNN, SVM) and deep learning (ResNet-50, VGG-16, Custom CNN) classification models to histopathological pictures and statistical data in order to determine the type of tumour the patient has: benign-noncancerous or malignant-cancerous.

The research would assist us in determining the optimal model for this type of classification. Further research might be conducted based on this comparison, and eventually, a fully functional Breast cancer detection system could be built for use in the medical industry.

Chapter 02: Feasibility Study, Requirements Analysis and Design

2.1 Feasibility Study

2.1.1 Problem Definition

Histopathological imaging of the breast tumour provides a microscopic cross-sectional view of the breast tissue. The tumour, on the other hand, was not extracted from the microscopic scans. As a result, image processing is required to identify the severity of the tumour, which is based on its size and location. Because there are many different forms of breast tumours, and manual detection and classification for a large amount of data can be highly inaccurate, automated tumour detection and classification is regarded highly advantageous.

2.1.2 Problem Analysis

An examination of the problem statement was carried out using a literature review and the previously done related works. An extensive literature review aided in a better understanding of past work, approaches and algorithms employed, and performance indicators employed. The main goal of the literature review was to see what was available in the scholarly literature on the subject, and a literature survey of numerous journal articles would help us figure out what models were constructed and what research findings came from them, as well as compare and contrast them. This comparison model aims to provide a better understanding of all the types of machine and deep learning based classification models to help us classify the authenticity of the presence of tumour , as a result, eventually, a fully functional Breast cancer detection system could be built for use in the medical industry.

Related Work:

Boosting Breast Cancer Detection Using Convolutional Neural Network by Saad Awadh Alanazi, M. M. Kamruzzaman,MdNazirul Islam Sarker,MadallahAlruwaili,YousefAlhwaiti,NasserAlshammari, and Muhammad Hameed Siddiqi **et al.2021**

For the automatic identification of breast cancer, the study developed a CNN approach that evaluates the IDC tissue areas in WSIs. In this study, three distinct CNN architectures were described, along with a proper comparison. The proposed approach, which employed CNN Model 3, achieved an accuracy of 87 percent. Model 3's five-layer CNN was best suited for this task, despite the fact that it was deeper than Models 1 and 2. A large collection of roughly 275,000 50 50-pixel RGB picture patches guided all structures. On comparing the suggested model to the machine learning (ML) algorithm, it was found that the proposed model outperformed the algorithm by 8%. The proposed model was proven to produce accurate findings, which could eliminate human error in the diagnosis process and lower the cost of a cancer diagnosis. The use of a secondary database like Kaggle is the study's biggest weakness, and future studies should be based on primary data for more accurate breast cancer detection outcomes.

Convolutional neural network-based models for diagnosis of breast cancer by Mehedi Masud, Amr E. Eldin Rashed & M. Shamim Hossain **et al.2020-**

This work used transfer learning to observe the classification performance of breast cancer from ultrasound pictures using eight pre-trained CNN models with fine-tuning. The photos were integrated from two separate datasets, and the Adam, RMSprop, and SGDM optimizers were used to evaluate the fine-tuned pre-trained models. The ResNet50 with Adam optimizer had the highest accuracy of 92.4 percent, and VGG16 had the highest AUC 0.97 score. It also suggested a shallow custom model because the pre-trained models had not produced the expected results, and all of the pre-trained models had many convolutional layers and required a long training period. As feature extractors, the proposed custom model used only one convolutional layer. The custom model was 100 percent accurate and had an AUC value of 1.0. In terms of training time, the custom model outperformed all other models and necessitated a small number of trainable parameters. The model was to be validated with other datasets, including new ultrasound pictures, in the future.

Using Three Machine Learning Techniques for Predicting Breast Cancer Recurrence by Ahmad LG*, Eshlaghy AT, Pourebrahimi A, Ebrahimi M and Razavi AR Department of Management Information Systems, Science and Research Branch, Islamic Azad University of Tehran-Iran, Iran

There were 1189 records in the dataset, 22 predictor variables, and one outcome variable. To create the predictive models, the researchers used machine learning approaches such as Decision Tree (C4.5), Support Vector Machine (SVM), and Artificial Neural Network (ANN). The major purpose of this work was to examine the sensitivity, specificity, and accuracy of these three well-known algorithms on the data. The accuracy of the DT, ANN, and SVM was 0.936, 0.947, and 0.957, respectively, according to the analysis. With the lowest error rate and maximum accuracy, the SVM classification model predicted breast cancer recurrence. The DT model's anticipated accuracy was the lowest of all. The results were obtained by utilising 10-fold cross-validation to assess each model's unbiased prediction accuracy.

SVM and SVM Ensembles in Breast Cancer Prediction BY Min-Wei Huang,Chih-Wen Chen ,Wei-Chao Lin,Shih-Wen Ke,Chih-Fong Tsai **et al. January 6, 2017**
https://doi.org/10.1371/journal.pone.0161501

The goal of this study was to evaluate the efficacy of SVM and SVM ensembles in predicting breast cancer outcomes on small and big size datasets. Training SVM and SVM ensembles were compared in terms of classification accuracy, ROC, F-measure, and computational durations. The experimental results demonstrated that for a small-scale dataset, linear kernel based SVM ensembles based on the bagging method and RBF kernel based SVM ensembles using the boosting method were preferable alternatives,

and feature selection must be done during the data pre-processing step. RBF kernel-based SVM ensembles based on boosting outperformed the other classifiers on a large-scale dataset.

BREAST CANCER CLASSIFICATION USING K-NEAREST NEIGHBOURS ALGORITHM Can EYUBOGLU Istanbul Commerce University, Department of Computer Engineering, Istanbul-Turkey ceyupoglu@ticaret.edu.tr

The k-Nearest Neighbours algorithm (k-NN) was used to classify breast cancer illness in this investigation. Furthermore, k-NN was implemented for various k values, and the resulting classification accuracies were compared. Breast cancer disease was successfully classified using k-NN, according to the study's findings. To evaluate the success of k-NN, the classification accuracies and error values were obtained. The acquired classification accuracy of k-NN was roughly 97 percent, according to the test findings. Furthermore, the findings of the study suggest that k-NN is a good classifier for classifying breast cancer disease.

Breast Cancer Detection Based on Feature Selection Using Support Vector Machine Algorithms Sunil Kumar, Directorate of Livestock Farm Guru Angad Dev Veterinary and Animal Science University Ludhiana, Punjab, India | Maninder Singh Department of Computer Science Punjabi University, Patiala, Punjab, India **et al.5 November, 2020**

On the standard benchmark, Wisconsin Diagnostic Breast Cancer (WDBC) database retrieved from the UCI Machine Learning Repository, the performance of the proposed SVM technique had been validated. The WDBC dataset contains 569 records (one record for each patient), 357 of which are classified as benign breast cancer patients and the rest as malignant breast cancer patients. Prior to feature selection, data pre-processing in the form of data normalisation was undertaken in this investigation. 10-fold cross-validation was used to produce accurate and impartial classification findings. The dataset was divided into 10 equal sections in a 10-fold CV. The SVM classifier was trained using nine parts (train set) in each fold, and the classification accuracy for the tenth part was calculated (test set) . This procedure was performed ten times, with each portion being classed as a test set. The SVM classifier utilised in this study was a two-class classifier, which classified the selected subset of characteristics as benign or malignant breast cancer tumours. A confusion matrix containing information about the classification system's actual and expected classifications was then constructed. Using 10-fold cross validation, the proposed method reduced the feature dimensionality from 30 to just six, resulting in a classification accuracy of 98.24 percent.

Breast Cancer Diagnosis Using Image Processing and Machine Learning by Subham Sadhukhan, Nityasree Upadhyay & Prerana Chakraborty conference paper First Online: 17 July 2019 et al. 17 July, 2019

The research described a computerised system for detecting cancer in its early stages in a short amount of time. The researchers used machine learning to train a model based on the predicted features of cell nuclei.. The accuracy of each classifier was tested in a comparison study of two separate methods, KNN and SVM. Following that, image processing was used to analyse a digital image of a fine needle aspirate (FNA) of breast tissue in order to determine the nuclei of the cells. The trained model's feature values were then used to determine whether the tumour produced was benign or malignant.

Deep Residual Learning for Image Recognition by Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun et al.2021-

The paper presented a residual learning framework for making it easier to train networks that are significantly deeper than those previously used. Instead of learning unreferenced functions, the layers

were explicitly created as learning residual functions with reference to the layer inputs. The empirical evidence showed that residual networks were easier to optimise and could benefit from significantly increased depth. On the ImageNet dataset, residual nets with up to 152 layers—8 are deeper than VGG nets but have lower complexity. On the ImageNet test set, an ensemble of these residual nets achieves a 3.57 per cent error. This result took first place in the classification task at the ILSVRC 2015.

2.1.3 Solution

2.2 Data-Flow Diagram (DFD)

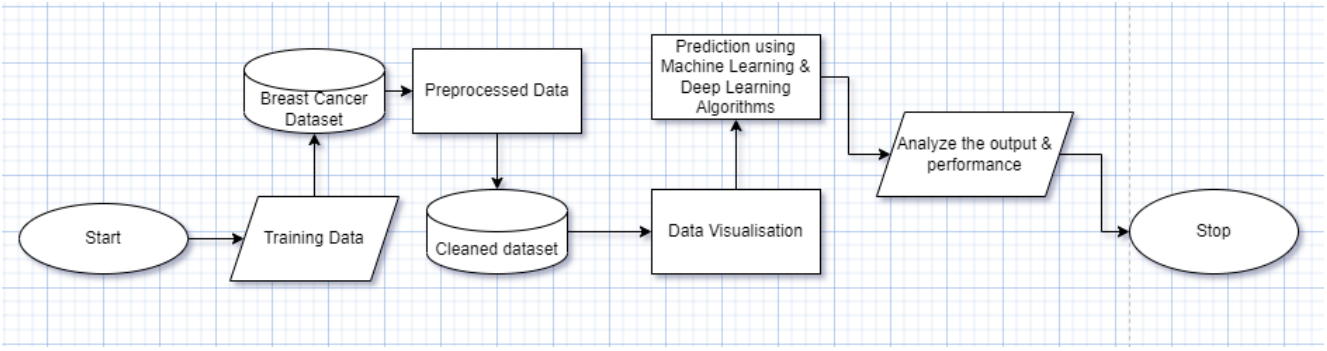


Figure 1 : Data Flow Diagram

**Flow chart drawn using draw.io [https://app.diagrams.net/]*

Chapter 03: IMPLEMENTATION

3.1 Date Set Used in the Minor Project

The following two datasets have been used in the project**:**

- **Wisconsin Breast Cancer Dataset**

[https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data]

Dataset Description:

Features are computed from a digitised image of a fine needle aspirate (FNA) of a breast mass. They describe the characteristics of the cell nuclei present in the image.

The 3-dimensional space is described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34].

Attribute Information:

1) ID number

2) Diagnosis (M = malignant, B = benign)

3) Ten real-valued features are computed for each cell nucleus:

- a) radius (mean of distances from centre to points on the perimeter)
- b) texture (standard deviation of grey-scale values)
- c) perimeter
- d) area
- e) smoothness (local variation in radius lengths)
- f) compactness (perimeter^2 / area - 1.0)
- g) concavity (severity of concave portions of the contour)
- h) concave points (number of concave portions of the contour)
- i) symmetry
- j) fractal dimension ("coastline approximation" - 1)

The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features.

For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

All feature values are recorded with four significant digits.

Missing attribute values: none

Class distribution: 357 benign, 212 malignant

- **Breast Histopathology Dataset**

https://www.kaggle.com/datasets/paultimothymooney/breast-histopathology-images

Dataset Description:

The original dataset consisted of 162 whole mount slide images of Breast Cancer (BCa) specimens scanned at 40x. From that, 277,524 patches of size 50 x 50 were extracted (198,738 IDC negative and 78,786 IDC positive). Each patch's file name is of the format: u*x*X*y*Y*classC.png* — > *example 10253*idx5*x1351*y1101*class0.png* . *Where u is the patient ID (10253*idx5), X is the x-coordinate of where this patch was cropped from, Y is the y-coordinate of where this patch was cropped from, and C indicates the class where 0 is non-IDC and 1 is IDC.
The original files are located here:

http://gleason.case.edu/webdata/jpi-dl-tutorial/IDC_regular_ps50_idx5.zip

Citation: https://www.ncbi.nlm.nih.gov/pubmed/27563488 and

http://spie.org/Publications/Proceedings/Paper/10.1117/12.2043872

3.2 Date Set Features

3.2.1 Types of Data Set

Two types of datasets were used for the proposed comparison model, one of them included the structured .csv data and the other included the histopathological images of the breast tissue. The dataset was then divided into Training and testing sections for the necessary validation. A further detailed description of the dataset for the binary classification has been provided above as well.

3.2.2 Number of Attributes, fields, description of the data set

The number of classes in this dataset are 2, for the presence of cancerous and non-cancerous tissue in the breast. Malignant or carcinoma and benign or non cancerous. The major two types of breast cancers : Ductal or lobular carcinoma is a type of cancer that affects the intestines.
The majority of breast cancers are carcinomas, which are tumours that begin in the epithelial cells that line the organs and tissues in the body. Adenocarcinoma, which begins in cells in the ducts (milk ducts) or the lobules, is the most common type of carcinoma that forms in the breast (glands in the breast that make milk).
In situ vs. invasive breast cancers : The type of breast cancer can also refer to whether the cancer has spread or not. In situ breast cancer (ductal carcinoma in situ or DCIS) is a pre-cancer that starts in a milk duct and has not grown into the rest of the breast tissue. The term invasive (or infiltrating) breast cancer is used to describe any type of breast cancer that has spread (invaded) into the surrounding breast tissue.

3.3 Design of Problem Statement

The severity of the situation was taken into consideration when creating the problem statement. Breast tumours are deemed fatal because they can push on healthy breast tissue or spread to other parts of the body. Some breast tumours are cancerous or may become cancerous in the future. If they restrict the flow of fluid surrounding the breast tissue, resulting in a rise in pressure inside the breast region, they can create difficulties. As a result, early detection and categorization are critical, and models like these, which are constructed using various algorithms, assist in precisely determining them.

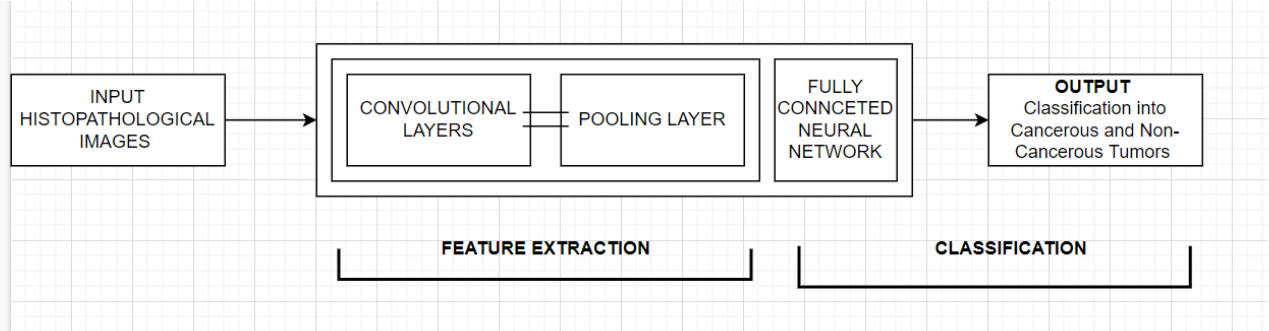


Figure 2 : Design of Problem Statement

**Flow chart drawn using draw.io [https://app.diagrams.net/]*

3.4 Algorithm / Pseudo code of the Project Problem

Machine Learning:

KNN:

1. The scikit-learn package is used to import the k-nearest neighbour algorithm.
2. Construct a feature and a target variable.
3. Split the data into training and testing sets.
4. Create a k-NN model with the neighbours value.
5. The data is trained or fitted into the model.
6. Predict the result.
7. Using the scoring method, determine the model's accuracy.

SVM:

1. The scikit-learn package is used to import the SVC (Support vector Classifier)algorithm.
2. Construct a feature and a target variable.
3. Split the data into training and testing sets.
4. Create a Linear SVC object.
5. Using the training data,train the linear SVC classifier.
6. Predict the result.
7. Using the scoring method,check the accuracy of the model.

Deep Learning:

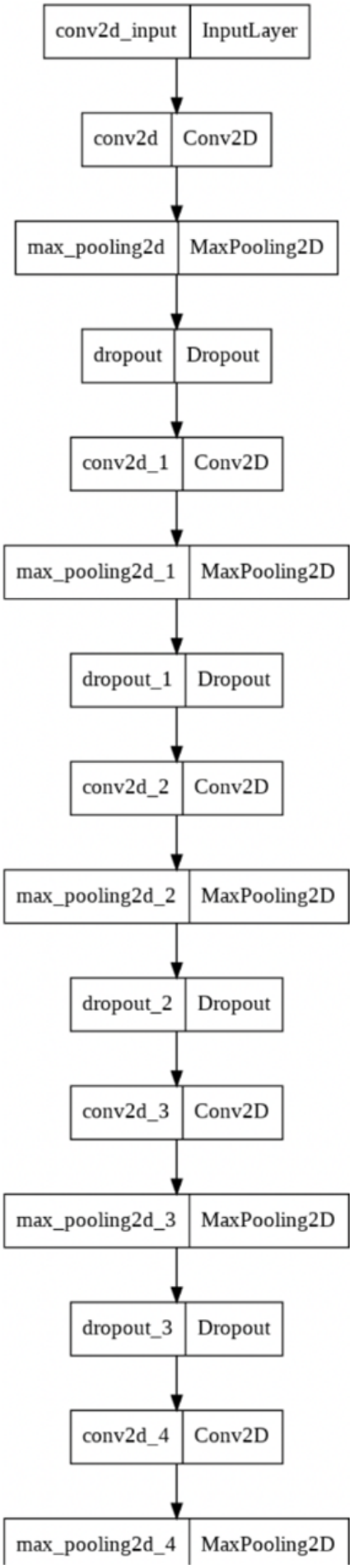
Pseudo code 1 Pre Processing:

```
1. Func create_training_data():
2. class_num <- category_index
3.       For image in images
4.                img_array <- read_image
5.                new_array <- resize_image(img_array)
6.                training_data <- append(new_array,class_num)
```

Pseudo code 2 Model Building :

1. model <- init_sequential()
2. Add input layer
3. Add convolution layer, set filters(64 or 128 or 256), kernel size(3x3 or 5x5),etc
4. Add max pooling layer, set pool size(2x2), etc
5. Add dropout layer, set the dropout percent
6. Repeat steps 3-5 as required
7. Flatten the output of these layers
8. Add dense and dropout layers
9. Get classification as output

2.5 Flow graph of the Minor Project Problem


conv2d_input InputLayer
conv2d Conv2D
max_pooling2d MaxPooling2D
dropout Dropout
conv2d_1 Conv2D
max_pooling2d_1 MaxPooling2D
dropout_1 Dropout
conv2d_2 Conv2D
max_pooling2d_2 MaxPooling2D
dropout_2 Dropout
conv2d_3 Conv2D
max_pooling2d_3 MaxPooling2D
dropout_3 Dropout
conv2d_4 Conv2D
max_pooling2d_4 MaxPooling2D

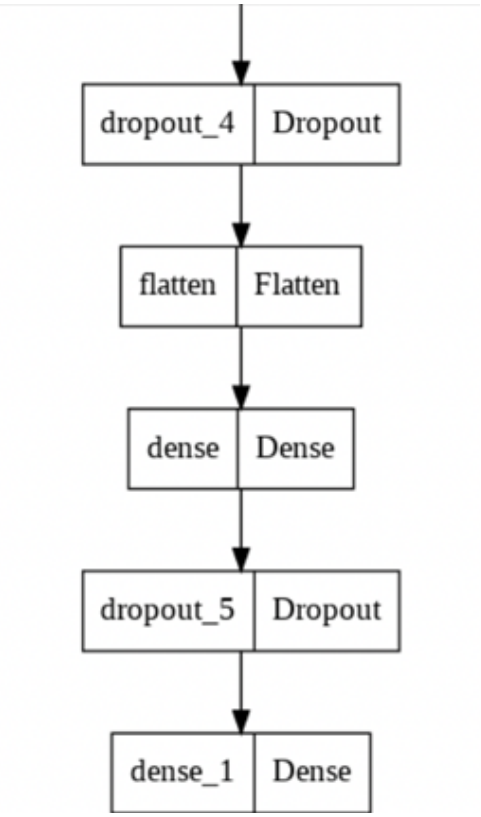


Figure 3 : Flow of the Model

3.6 Screenshots of the various stages of Project

Machine Learning:

Data Preprocessing :

```
#import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files  No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving bs.csv to bs (1).csv
```

```
[7]  df = pd.read_csv("bs.csv")
```

Figure 4 : Importing all the necessary libraries and dataset

```
[11]  df = df.dropna(axis=1)
```

```
[12]  df['diagnosis'].value_counts()

B    357
M    212
Name: diagnosis, dtype: int64
```

Figure 5 : Dropping Null Values

```
[18] X = df.iloc[:, 2:31].values
     Y = df.iloc[:, 1].values
```

```
[19] from sklearn.model_selection import train_test_split
     X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, random_state = 0)
```

Figure 6: Train Test split

SVM(Support Vector Machine):

```
[24] from sklearn.svm import SVC
     svc_model = SVC()
```

```
[26] svc_model.fit(X_train, Y_train)

     SVC()
```

```
[27] y_predict = svc_model.predict(X_test)
```

```
     from sklearn.metrics import accuracy_score
     accuracy_score(Y_test,y_predict)

     0.9370629370629371
```

Figure 7 : SVM model

KNN(K-Nearest Neighbor):

Using Sklearn

```
[ ]  from sklearn.neighbors import KNeighborsClassifier
     knn = KNeighborsClassifier(n_neighbors = 13)
     knn.fit(X_train, y_train)

     KNeighborsClassifier(n_neighbors=13)
```

Prediction Score

```
[ ]  knn.score(X_test, y_test)

     0.9627659574468085
```

Figure 8 : KNN model

Deep Learning:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import os
from sklearn.metrics import confusion_matrix
import itertools

from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from tensorflow.keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau
from sklearn.metrics import accuracy_score, classification_report
from keras.models import model_from_json
```

Figure 9 : Importing all the necessary libraries

```python
[ ] from google.colab import files
    files.upload()
```

```
Choose Files  No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving kaggle.json to kaggle (1).json
{'kaggle.json': b'{"username":"kunikasharma191227","key":"d090de9dd729e73d79666c0454a9653b"}'}
```

```python
[ ] !mkdir -p ~/.kaggle
    !cp "kaggle.json" ~/.kaggle/
    !cat ~/.kaggle/kaggle.json
    !chmod 600 ~/.kaggle/kaggle.json
    !kaggle datasets download -d paultimothymooney/breast-histopathology-images
```

```
{"username":"kunikasharma191227","key":"d090de9dd729e73d79666c0454a9653b"}breast-histopathology-images.zip: Skipping, found more recently m
```

Figure 10 : Importing the dataset from Kaggle using API

```python
!unzip breast-histopathology-images.zip
```

```
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_y1601_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_y1651_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_y1701_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_y1751_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_y1801_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_y1851_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_y1901_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_y1951_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_y2001_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_y2251_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_y2301_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_y2351_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2351_y2401_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_y1101_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_y1151_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_y1201_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_y1251_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_y1301_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_y1351_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_y1401_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_y1451_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_y1501_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_y1551_class0.png
inflating: IDC_regular_ps50_idx5/9346/0/9346_idx5_x2401_y1601_class0.png
```

Figure 11: Unzipping dataset

1. Custom based CNN Model

```
DATADIR = r"/content/10264"
CATEGORIES = ["0","1"]
for category in CATEGORIES:
    path = os.path.join(DATADIR,category)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path,img))
        plt.imshow(img_array)
        plt.show()
        plt.axis("off")
        break
    break
```

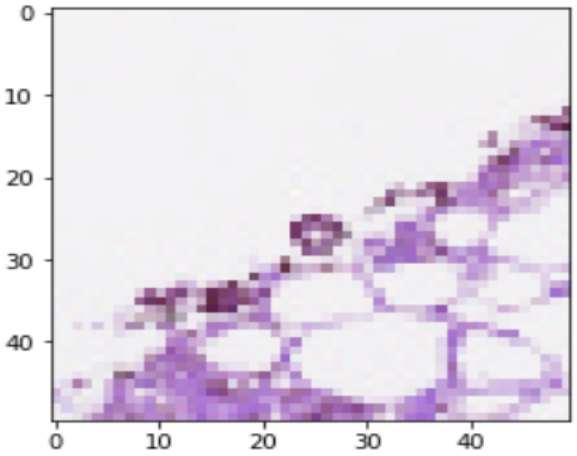


Figure 12: Displaying a picture from the dataset

```
IMG_SIZE = 150
new_array = cv2.resize(img_array,(IMG_SIZE,IMG_SIZE))
plt.imshow(new_array,cmap = "gray")
plt.axis("off")

```

```
(-0.5, 149.5, 149.5, -0.5)
```

Figure 13 : Resizing the previously displayed image

```
training_data = []

def create_training_data():
    for category in CATEGORIES:
        path = os.path.join(DATADIR,category)
        class_num = CATEGORIES.index(category)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
                new_array = cv2.resize(img_array,(IMG_SIZE,IMG_SIZE))
                training_data.append([new_array,class_num])
            except Exception as e:
                pass
create_training_data()
```

```
X = []
y = []
for features,label in training_data:
    X.append(features)
    y.append(label)
X = np.array(X).reshape(-1,IMG_SIZE,IMG_SIZE)
print(X.shape)
X = X/255.0
X = X.reshape(-1,150,150,1)
```

```
(1204, 150, 150)
```

Figure 14 : Training the dataset and converting the image from RGB to Grayscale

```
model = Sequential()

model.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',
                 activation ='relu', input_shape = (150,150,1)))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters = 128, kernel_size = (3,3),padding = 'Same',
                 activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters = 128, kernel_size = (3,3),padding = 'Same',
                 activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.3))

model.add(Conv2D(filters = 128, kernel_size = (2,2),padding = 'Same',
                 activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.3))


model.add(Conv2D(filters = 256, kernel_size = (2,2),padding = 'Same',
                 activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.3))
```

Figure 15 : Creating the CNN model framework

```
[ ]  epochs = 15
     batch_size = 40
```

Figure 16 : Setting the epoch and batch size

```
Epoch 1/40
24/24 [==============================] - 117s 5s/step - loss: 0.8276 - accuracy: 0.4865 - val_loss: 0.9694 - val_accuracy: 0.4274
Epoch 2/40
24/24 [==============================] - 111s 5s/step - loss: 0.6739 - accuracy: 0.5504 - val_loss: 0.8543 - val_accuracy: 0.4274
Epoch 3/40
24/24 [==============================] - 111s 5s/step - loss: 0.4969 - accuracy: 0.8277 - val_loss: 1.4134 - val_accuracy: 0.4274
Epoch 4/40
24/24 [==============================] - 110s 5s/step - loss: 0.3464 - accuracy: 0.8765 - val_loss: 1.0824 - val_accuracy: 0.5602
Epoch 5/40
24/24 [==============================] - 110s 5s/step - loss: 0.3103 - accuracy: 0.8971 - val_loss: 1.4675 - val_accuracy: 0.4647
Epoch 6/40
24/24 [==============================] - 113s 5s/step - loss: 0.2829 - accuracy: 0.9101 - val_loss: 1.3255 - val_accuracy: 0.5643
Epoch 7/40
24/24 [==============================] - 113s 5s/step - loss: 0.4663 - accuracy: 0.7551 - val_loss: 1.2633 - val_accuracy: 0.5145
Epoch 8/40
24/24 [==============================] - 111s 5s/step - loss: 0.3163 - accuracy: 0.8852 - val_loss: 1.4919 - val_accuracy: 0.4772
Epoch 9/40
24/24 [==============================] - 111s 5s/step - loss: 0.3031 - accuracy: 0.9014 - val_loss: 0.8843 - val_accuracy: 0.6473
Epoch 10/40
24/24 [==============================] - 110s 5s/step - loss: 0.2839 - accuracy: 0.9057 - val_loss: 1.0174 - val_accuracy: 0.6224
Epoch 11/40
24/24 [==============================] - 114s 5s/step - loss: 0.2585 - accuracy: 0.9177 - val_loss: 1.2165 - val_accuracy: 0.6100
Epoch 12/40
24/24 [==============================] - 110s 5s/step - loss: 0.2567 - accuracy: 0.9155 - val_loss: 1.3050 - val_accuracy: 0.5685
Epoch 13/40
24/24 [==============================] - 109s 5s/step - loss: 0.3512 - accuracy: 0.8570 - val_loss: 1.1402 - val_accuracy: 0.5685
Epoch 14/40
24/24 [==============================] - 109s 5s/step - loss: 0.2677 - accuracy: 0.9198 - val_loss: 1.3417 - val_accuracy: 0.5311
Epoch 15/40
```

```
Epoch 28/40
24/24 [==============================] - 109s 5s/step - loss: 0.2224 - accuracy: 0.9231 - val_loss: 0.8925 - val_accuracy: 0.6266
Epoch 29/40
24/24 [==============================] - 109s 5s/step - loss: 0.2259 - accuracy: 0.9242 - val_loss: 0.7613 - val_accuracy: 0.6846
Epoch 30/40
24/24 [==============================] - 109s 5s/step - loss: 0.2126 - accuracy: 0.9274 - val_loss: 0.9396 - val_accuracy: 0.6266
Epoch 31/40
24/24 [==============================] - 109s 5s/step - loss: 0.2524 - accuracy: 0.9090 - val_loss: 0.8100 - val_accuracy: 0.6307
Epoch 32/40
24/24 [==============================] - 109s 5s/step - loss: 0.2047 - accuracy: 0.9285 - val_loss: 0.7698 - val_accuracy: 0.6598
Epoch 33/40
24/24 [==============================] - 110s 5s/step - loss: 0.2252 - accuracy: 0.9187 - val_loss: 0.8806 - val_accuracy: 0.6390
Epoch 34/40
24/24 [==============================] - 109s 5s/step - loss: 0.2593 - accuracy: 0.9025 - val_loss: 0.8456 - val_accuracy: 0.6058
Epoch 35/40
24/24 [==============================] - 110s 5s/step - loss: 0.1993 - accuracy: 0.9285 - val_loss: 0.7456 - val_accuracy: 0.6722
Epoch 36/40
24/24 [==============================] - 109s 5s/step - loss: 0.2170 - accuracy: 0.9263 - val_loss: 0.8056 - val_accuracy: 0.6473
Epoch 37/40
24/24 [==============================] - 109s 5s/step - loss: 0.2108 - accuracy: 0.9328 - val_loss: 0.5259 - val_accuracy: 0.7344
Epoch 38/40
24/24 [==============================] - 109s 5s/step - loss: 0.2263 - accuracy: 0.9220 - val_loss: 0.4375 - val_accuracy: 0.7884
Epoch 39/40
24/24 [==============================] - 109s 5s/step - loss: 0.3435 - accuracy: 0.8602 - val_loss: 0.4483 - val_accuracy: 0.7510
Epoch 40/40
24/24 [==============================] - 114s 5s/step - loss: 0.2324 - accuracy: 0.9146 - val_loss: 0.7679 - val_accuracy: 0.6971
```

Figure 17-18 Training the model

```
[ ]  1  loaded_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
     2  score = loaded_model.evaluate(X_val, Y_val, verbose=0)
     3  print("%s: %.2f%%" % (loaded_model.metrics_names[1], score[1]*100))
```

```
accuracy: 69.71%
```

Figure 19: Validation on Performance metric ACCURACY

```
[ ]  # serialize model to JSON
     from keras.models import model_from_json
     model_json = model.to_json()
     with open("model.json", "w") as json_file:
         json_file.write(model_json)
```

```
[ ]  # serialize weights to HDF5
     model.save_weights("model.h5")
     print("Saved model to disk")
```

```
Saved model to disk
```

```
     json_file = open('/content/model.json', 'r')
     loaded_model_json = json_file.read()
     json_file.close()
     loaded_model = model_from_json(loaded_model_json)

     loaded_model.load_weights("/content/model.h5")
     print("Loaded model from disk")
```

```
Loaded model from disk
```

Figure 20: Saving the model

2. Resnet-50 Pretrained model

```
data = '/content/breast-histopathology-images'
No_breast_cancer = '/content/breast-histopathology-images/10264/0'
Yes_breast_cancer = '/content/breast-histopathology-images/10264/1'
```

```
dirlist=[No_breast_cancer, Yes_breast_cancer]
classes=['No', 'Yes']
filepaths=[]
labels=[]
for i,j in zip(dirlist, classes):
    filelist=os.listdir(i)
    for f in filelist:
        filepath=os.path.join (i,f)
        filepaths.append(filepath)
        labels.append(j)
print ('filepaths: ', len(filepaths), '   labels: ', len(labels))
```

```
filepaths:  1204    labels:  1204
```

```
Files=pd.Series(filepaths, name='filepaths')
Label=pd.Series(labels, name='labels')
df=pd.concat([Files,Label], axis=1)
df=pd.DataFrame(np.array(df).reshape(1204,2), columns = ['filepaths', 'labels'])
df.head()
```

	filepaths	labels
0	/content/breast-histopathology-images/10264/0/...	No
1	/content/breast-histopathology-images/10264/0/...	No
2	/content/breast-histopathology-images/10264/0/...	No
3	/content/breast-histopathology-images/10264/0/...	No
4	/content/breast-histopathology-images/10264/0/...	No

Figure 21: Data Segmentation

```
from sklearn.model_selection import train_test_split
train, test = train_test_split(df, train_size=0.95, random_state=0)
train_new, valid = train_test_split(train, train_size=0.90, random_state=0)
```

```
print(f"train set shape: {train_new.shape}")
print(f"test set shape: {test.shape}")
print(f"validation set shape: {valid.shape}")
```

```
train set shape: (1028, 2)
test set shape: (61, 2)
validation set shape: (115, 2)
```

Figure 22: Train-Test Split

```
train_datagen = ImageDataGenerator(rescale = 1./255.,rotation_range = 40, width_shift_range = 0.2, height_shift_range = 0.2,
                                   shear_range = 0.2, zoom_range = 0.2, horizontal_flip = True, vertical_flip =True)
test_datagen = ImageDataGenerator(rescale = 1.0/255.)
```

```
train_gen = train_datagen.flow_from_dataframe(dataframe = train_new,
                                              x_col = 'filepaths', y_col ='labels',
                                              target_size = (224,224), batch_size = 32,
                                              class_mode = 'binary', shuffle = True)
val_gen = train_datagen.flow_from_dataframe(valid,
                                            target_size=(224,224), x_col = 'filepaths', y_col ='labels',
                                            class_mode='binary',
                                            batch_size= 16, shuffle=True)
test_gen = test_datagen.flow_from_dataframe(test,
                                            target_size = (224,224), x_col = 'filepaths', y_col ='labels',
                                             class_mode = 'binary',
                                            batch_size = 16, shuffle = False)
```

```
Found 1028 validated image filenames belonging to 2 classes.
Found 115 validated image filenames belonging to 2 classes.
Found 61 validated image filenames belonging to 2 classes.
```

Figure 23: Data Augmentation

```
from tensorflow import keras
base_model = keras.applications.ResNet50V2(
    weights="imagenet",  # Load weights pre-trained on ImageNet.
    input_shape=(224, 224, 3),
    include_top=False,
)  # Do not include the ImageNet classifier at the top.
base_model.trainable = False
inputs = keras.Input(shape=(224, 224, 3))
x = base_model(inputs, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)
x = keras.layers.Dropout(0.2)(x)  # Regularize with dropout
outputs = keras.layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50v2_weights_tf_dim_ordering_tf_kernels_notop.h5
94674944/94668760 [==============================] - 1s 0us/step
94683136/94668760 [==============================] - 1s 0us/step
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 resnet50v2 (Functional)     (None, 7, 7, 2048)        23564800

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 2048)              0

 dense (Dense)               (None, 1)                 2049

=================================================================
Total params: 23,566,849
Trainable params: 2,049
Non-trainable params: 23,564,800
_________________________________________________________________
```

Figure 24: CNN(Resnet50) model

```
callbacks = [tf.keras.callbacks.ModelCheckpoint("Tumor_classifier_model.h5", save_best_only=True, verbose = 0)]
model.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate= 0.0001), metrics=['accuracy'])
history = model.fit(train_gen, validation_data = val_gen, epochs = 50, callbacks = [callbacks], verbose = 1)
```

```
Epoch 1/50
33/33 [==============================] - 175s 5s/step - loss: 0.7408 - accuracy: 0.5195 - val_loss: 0.6638 - val_accuracy: 0.5739
Epoch 2/50
33/33 [==============================] - 173s 5s/step - loss: 0.6929 - accuracy: 0.5720 - val_loss: 0.6606 - val_accuracy: 0.5826
Epoch 3/50
33/33 [==============================] - 171s 5s/step - loss: 0.6328 - accuracy: 0.6547 - val_loss: 0.5658 - val_accuracy: 0.7043
Epoch 4/50
33/33 [==============================] - 177s 5s/step - loss: 0.5944 - accuracy: 0.6955 - val_loss: 0.5203 - val_accuracy: 0.7913
Epoch 5/50
33/33 [==============================] - 176s 5s/step - loss: 0.5450 - accuracy: 0.7237 - val_loss: 0.5051 - val_accuracy: 0.8348
Epoch 6/50
33/33 [==============================] - 173s 5s/step - loss: 0.5292 - accuracy: 0.7558 - val_loss: 0.5118 - val_accuracy: 0.7913
Epoch 7/50
33/33 [==============================] - 174s 5s/step - loss: 0.4966 - accuracy: 0.7753 - val_loss: 0.4813 - val_accuracy: 0.7826
Epoch 8/50
33/33 [==============================] - 172s 5s/step - loss: 0.4792 - accuracy: 0.7967 - val_loss: 0.4525 - val_accuracy: 0.8522
Epoch 9/50
33/33 [==============================] - 178s 5s/step - loss: 0.4724 - accuracy: 0.7957 - val_loss: 0.4305 - val_accuracy: 0.8435
Epoch 10/50
33/33 [==============================] - 173s 5s/step - loss: 0.4443 - accuracy: 0.8210 - val_loss: 0.4355 - val_accuracy: 0.8348
Epoch 11/50
33/33 [==============================] - 172s 5s/step - loss: 0.4490 - accuracy: 0.8210 - val_loss: 0.4191 - val_accuracy: 0.8435
Epoch 12/50
33/33 [==============================] - 174s 5s/step - loss: 0.4161 - accuracy: 0.8298 - val_loss: 0.3732 - val_accuracy: 0.8957
Epoch 13/50
33/33 [==============================] - 173s 5s/step - loss: 0.4213 - accuracy: 0.8346 - val_loss: 0.3853 - val_accuracy: 0.8696
Epoch 14/50
33/33 [==============================] - 172s 5s/step - loss: 0.4100 - accuracy: 0.8375 - val_loss: 0.3542 - val_accuracy: 0.9130
Epoch 15/50
33/33 [==============================] - 172s 5s/step - loss: 0.3931 - accuracy: 0.8541 - val_loss: 0.3781 - val_accuracy: 0.8870
Epoch 16/50
33/33 [==============================] - 171s 5s/step - loss: 0.3865 - accuracy: 0.8531 - val_loss: 0.3690 - val_accuracy: 0.9043
Epoch 17/50
33/33 [==============================] - 171s 5s/step - loss: 0.3708 - accuracy: 0.8716 - val_loss: 0.3635 - val_accuracy: 0.8609
Epoch 18/50
33/33 [==============================] - 172s 5s/step - loss: 0.3720 - accuracy: 0.8580 - val_loss: 0.3462 - val_accuracy: 0.8957
Epoch 19/50
33/33 [==============================] - 172s 5s/step - loss: 0.3593 - accuracy: 0.8599 - val_loss: 0.3164 - val_accuracy: 0.9217
Epoch 20/50
33/33 [==============================] - 171s 5s/step - loss: 0.3600 - accuracy: 0.8716 - val_loss: 0.3298 - val_accuracy: 0.8957
Epoch 21/50
33/33 [==============================] - 171s 5s/step - loss: 0.3520 - accuracy: 0.8677 - val_loss: 0.3213 - val_accuracy: 0.9130
Epoch 22/50
33/33 [==============================] - 171s 5s/step - loss: 0.3491 - accuracy: 0.8755 - val_loss: 0.3346 - val_accuracy: 0.8870
Epoch 23/50
33/33 [==============================] - 172s 5s/step - loss: 0.3350 - accuracy: 0.8823 - val_loss: 0.3150 - val_accuracy: 0.9217
```

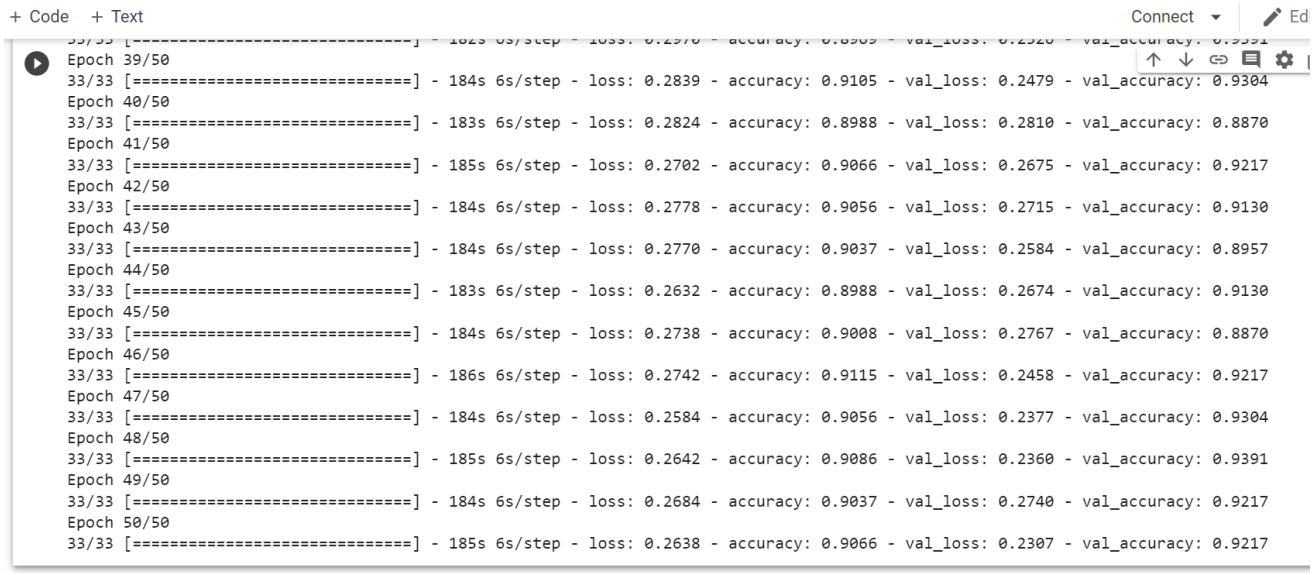


Figure 25: Training the model

```
[ ]  model.save("model.h5")
```

```
[ ] from PIL import Image
    model_path = "model.h5"
    loaded_model = tf.keras.models.load_model(model_path)
    import numpy as np

    image = cv2.imread("/content/breast-histopathology-images/10264/0/10264_idx5_x1001_y551_class0.png")

    image_fromarray = Image.fromarray(image, 'RGB')
    resize_image = image_fromarray.resize((224, 224))
    expand_input = np.expand_dims(resize_image,axis=0)
    input_data = np.array(expand_input)
    input_data = input_data/255

    pred = loaded_model.predict(input_data)
    if pred >= 0.5:
      print("Yes")
    else:
      print("No")

    No
```

Figure 26: Saving and Predicting result

3. VGG-16 Pre-trained Model

```
imagePatches = glob('/content/breast-histopathology-images/10264/**/*', recursive=True)
for filename in imagePatches[0:10]:
    print(filename)
```

```
/content/breast-histopathology-images/10264/0
/content/breast-histopathology-images/10264/1
/content/breast-histopathology-images/10264/0/10264_idx5_x2251_y351_class0.png
/content/breast-histopathology-images/10264/0/10264_idx5_x601_y2101_class0.png
/content/breast-histopathology-images/10264/0/10264_idx5_x2401_y1101_class0.png
/content/breast-histopathology-images/10264/0/10264_idx5_x1751_y501_class0.png
/content/breast-histopathology-images/10264/0/10264_idx5_x801_y751_class0.png
/content/breast-histopathology-images/10264/0/10264_idx5_x2351_y1151_class0.png
/content/breast-histopathology-images/10264/0/10264_idx5_x2051_y551_class0.png
/content/breast-histopathology-images/10264/0/10264_idx5_x2601_y951_class0.png
```

```
# Two arrays holding images by class type

class0 = [] # 0 = no cancer
class1 = [] # 1 = cancer

for filename in imagePatches:
    if filename.endswith("class0.png"):
         class0.append(filename)
    else:
        class1.append(filename)
```

```
len(class1+class0)
```

```
1206
```

```
sampled_class0 = random.sample(class0, 589)
sampled_class1 = random.sample(class1, 589)
len(sampled_class0+sampled_class0)
```

```
1178
```

```
img_size = 75
```

```
from matplotlib.image import imread
import cv2

def get_image_arrays(data, label):
    img_arrays = []
    for i in data:
        if i.endswith('.png'):
            img = cv2.imread(i ,cv2.IMREAD_COLOR)
            img_sized = cv2.resize(img, (img_size,img_size), interpolation=cv2.INTER_LINEAR)
            img_arrays.append([img_sized, label])
    return img_arrays
```

```
[11] class0_array = get_image_arrays(sampled_class0, 0)
     class1_array = get_image_arrays(sampled_class1, 1)
```

```
] combined_data = np.concatenate((class0_array, class1_array))
  random.seed(41)
  random.shuffle(combined_data)
```

```
<string>:6: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which
```

```
] X = []
  y = []

  for features,label in combined_data:
      X.append(features)
      y.append(label)
```

```
  # print(X[11].reshape(-1, 50, 50, 3))
  # reshape X data
  X = np.array(X).reshape(-1, img_size, img_size, 3)
```

+ Code + Text

```
] X.shape
```

```
(1176, 75, 75, 3)
```

Figure 27: Data Segmentation

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(882, 75, 75, 3) (294, 75, 75, 3) (882, 2) (294, 2)
```

Figure 28: Train-Test Split

```
from tensorflow.keras.applications.vgg16 import VGG16
```

```
base_model = VGG16(weights='imagenet', include_top=False,
                           input_shape=(img_size, img_size,3))

# freeze extraction layers
base_model.trainable = False

# add custom top layers
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.2)(x)
x = Dense(4096,activation="relu")(x)
x = Dense(4096,activation="relu")(x)
x = Dropout(0.2)(x)
x = Dense(2096,activation="relu")(x)
predictions = Dense(2, activation='sigmoid')(x)
model = Model(inputs=base_model.input, outputs=predictions)

# confirm unfrozen layers
for layer in model.layers:
    if layer.trainable==True:
        print(layer)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applicat
```

```
# Model Summary

model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 75, 75, 3)]       0

 block1_conv1 (Conv2D)       (None, 75, 75, 64)        1792

 block1_conv2 (Conv2D)       (None, 75, 75, 64)        36928

 block1_pool (MaxPooling2D)  (None, 37, 37, 64)        0

 block2_conv1 (Conv2D)       (None, 37, 37, 128)       73856

 block2_conv2 (Conv2D)       (None, 37, 37, 128)       147584

 block2_pool (MaxPooling2D)  (None, 18, 18, 128)       0

 block3_conv1 (Conv2D)       (None, 18, 18, 256)       295168

 block3_conv2 (Conv2D)       (None, 18, 18, 256)       590080

 block3_conv3 (Conv2D)       (None, 18, 18, 256)       590080
```

```
block3_pool (MaxPooling2D)  (None, 9, 9, 256)         0

block4_conv1 (Conv2D)       (None, 9, 9, 512)         1180160

block4_conv2 (Conv2D)       (None, 9, 9, 512)         2359808

block4_conv3 (Conv2D)       (None, 9, 9, 512)         2359808

block4_pool (MaxPooling2D)  (None, 4, 4, 512)         0

block5_conv1 (Conv2D)       (None, 4, 4, 512)         2359808

block5_conv2 (Conv2D)       (None, 4, 4, 512)         2359808

block5_conv3 (Conv2D)       (None, 4, 4, 512)         2359808

block5_pool (MaxPooling2D)  (None, 2, 2, 512)         0

global_average_pooling2d (G  (None, 512)              0
lobalAveragePooling2D)

dropout (Dropout)           (None, 512)               0

dense (Dense)               (None, 4096)              2101248

dense_1 (Dense)             (None, 4096)              16781312

dropout_1 (Dropout)         (None, 4096)              0
```

```
 dense_2 (Dense)             (None, 2096)              8587312

 dense_3 (Dense)             (None, 2)                 4194

=================================================================
Total params: 42,188,754
Trainable params: 27,474,066
Non-trainable params: 14,714,688
_________________________________________________________________
```

Figure 29: CNN(VGG-16) model

```
[22] callbacks = [EarlyStopping(monitor='val_loss', patience=5, verbose=1),
                  ModelCheckpoint('model.hdf5',
                                  save_best_only=True)]
```

```
[25] opt = Adam(learning_rate=0.0001)
     model.compile(
       loss='binary_crossentropy',
       optimizer=opt,
       metrics=['accuracy']
     )
```

```
%time
history=model.fit(X_train, y_train,validation_data=(X_test, y_test),verbose = 1,batch_size=32,epochs = 50,callbacks=callbacks)
```

```
CPU times: user 2 µs, sys: 1e+03 ns, total: 3 µs
Wall time: 7.15 µs
Epoch 1/50
28/28 [==============================] - 74s 3s/step - loss: 0.2274 - accuracy: 0.9127 - val_loss: 0.1298 - val_accuracy: 0.9490
Epoch 2/50
28/28 [==============================] - 73s 3s/step - loss: 0.1418 - accuracy: 0.9478 - val_loss: 0.1274 - val_accuracy: 0.9490
Epoch 3/50
28/28 [==============================] - 72s 3s/step - loss: 0.1393 - accuracy: 0.9512 - val_loss: 0.1891 - val_accuracy: 0.9490
Epoch 4/50
28/28 [==============================] - 78s 3s/step - loss: 0.1080 - accuracy: 0.9592 - val_loss: 0.0857 - val_accuracy: 0.9762
Epoch 5/50
28/28 [==============================] - 72s 3s/step - loss: 0.0785 - accuracy: 0.9705 - val_loss: 0.1346 - val_accuracy: 0.9558
Epoch 6/50
28/28 [==============================] - 71s 3s/step - loss: 0.1364 - accuracy: 0.9467 - val_loss: 0.1359 - val_accuracy: 0.9320
Epoch 7/50
28/28 [==============================] - 83s 3s/step - loss: 0.0700 - accuracy: 0.9762 - val_loss: 0.1362 - val_accuracy: 0.9558
Epoch 8/50
28/28 [==============================] - 85s 3s/step - loss: 0.0628 - accuracy: 0.9762 - val_loss: 0.1241 - val_accuracy: 0.9626
Epoch 9/50
28/28 [==============================] - 74s 3s/step - loss: 0.0453 - accuracy: 0.9875 - val_loss: 0.1151 - val_accuracy: 0.9796
Epoch 9: early stopping
```

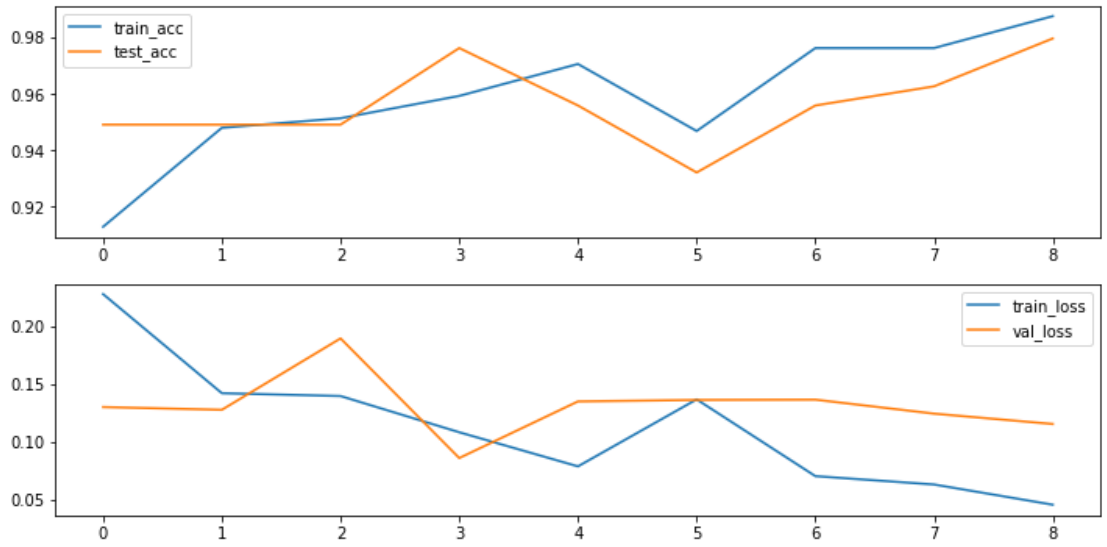
Figure 30: Training the model

```
def f1_score(y_true, y_pred): #taken from old keras source code
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    recall = true_positives / (possible_positives + K.epsilon())
    f1_val = 2*(precision*recall)/(precision+recall+K.epsilon())
    return f1_val
```

```
METRICS = [
      tf.keras.metrics.BinaryAccuracy(name='accuracy'),
      tf.keras.metrics.Precision(name='precision'),
      tf.keras.metrics.Recall(name='recall'),
      tf.keras.metrics.AUC(name='auc'),
        f1_score,
]
```

```
#plot the accuracy graph
plt.figure(figsize = (12,6))
plt.subplot(2,1,1)
plt.plot(history.history['accuracy'], label="train_acc")
plt.plot(history.history['val_accuracy'], label = "test_acc")
plt.legend()
plt.subplot(2,1,2)
plt.plot(history.history['loss'], label = "train_loss")
plt.plot(history.history['val_loss'], label = "val_loss")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fa1f77fddd0>
```

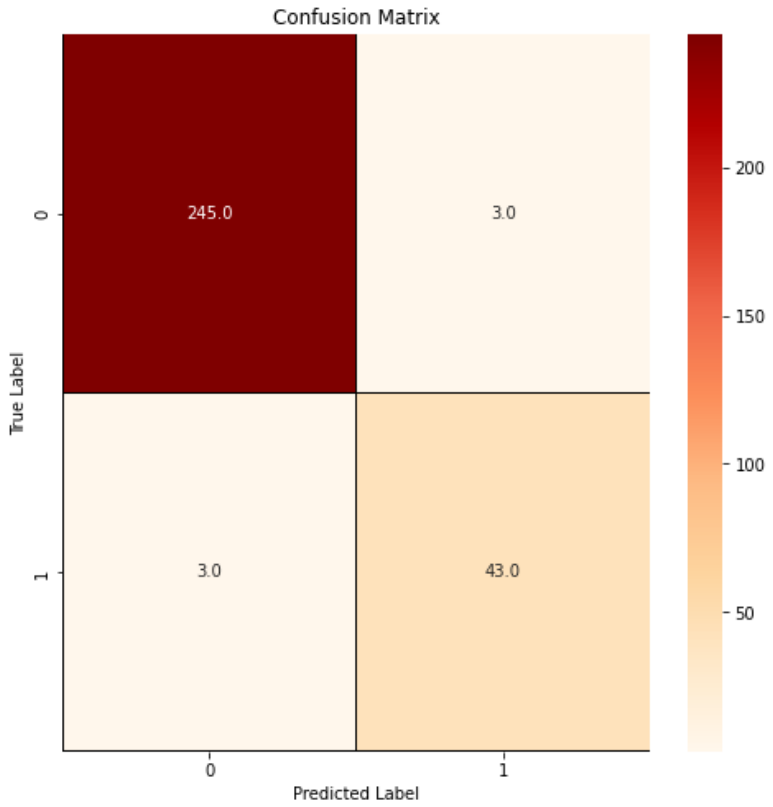



```
from sklearn.metrics import confusion_matrix
import seaborn as sns

Y_pred = model.predict(X_test)
Y_pred_classes = np.argmax(Y_pred,axis = 1)
Y_true = np.argmax(y_test,axis = 1)

confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)

f,ax = plt.subplots(figsize=(8, 8))
sns.heatmap(confusion_mtx, annot=True, linewidths=0.01,cmap="OrRd",linecolor="black", fmt= '.1f',ax=ax)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```




```
from sklearn.metrics import classification_report

print(classification_report(Y_true, Y_pred_classes))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       248
           1       0.93      0.93      0.93        46

    accuracy                           0.98       294
   macro avg       0.96      0.96      0.96       294
weighted avg       0.98      0.98      0.98       294
```

```
import pandas as pd
import matplotlib.pyplot as plt
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```

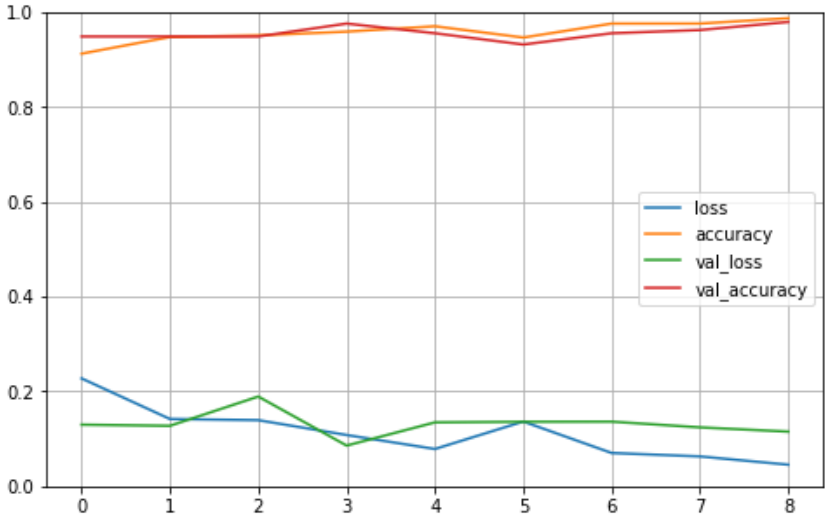


Figure 31: Result Analysis

```
[32] model.save("model.h5")
```

Figure 32: Saving Model

Chapter 04: RESULTS

4.1 Discussion on the Results Achieved

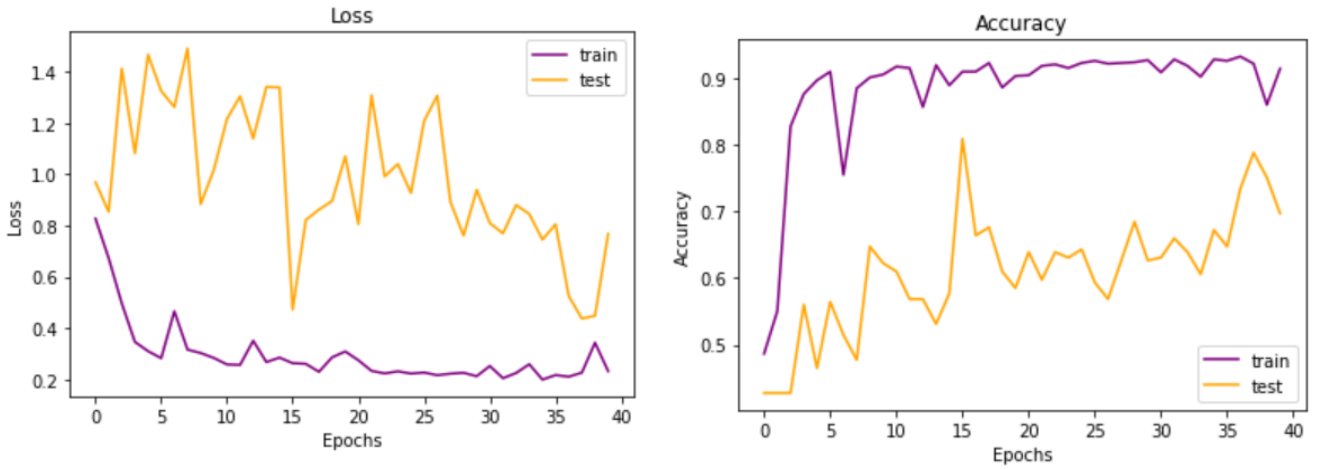


Figure 33-34 Accuracy Vs Loss curve for Custom CNN model

Resnet50:

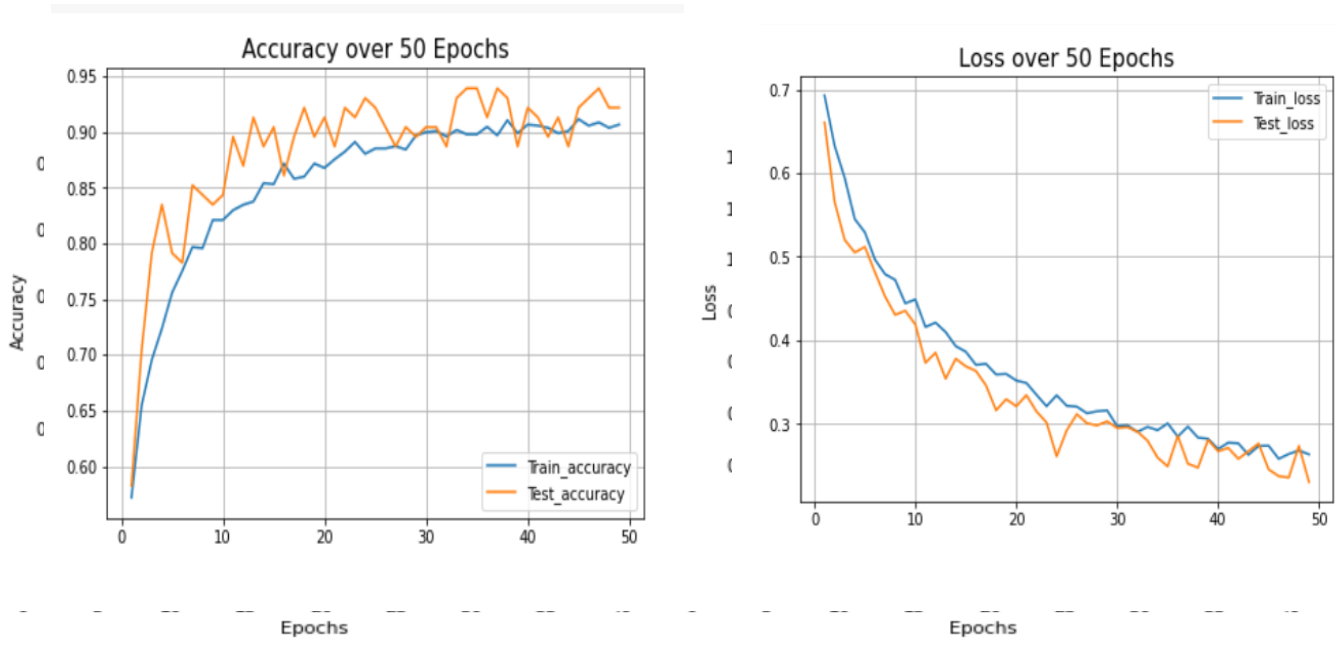


Figure 35-36 Accuracy Vs Loss curve for Resnet50 CNN model

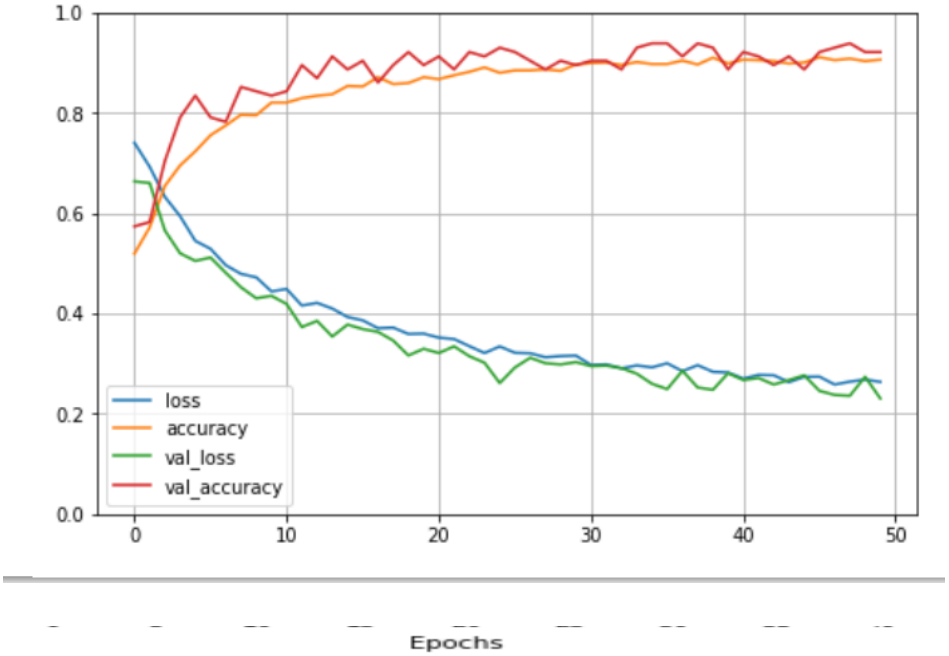


Figure 37: Training and validation Accuracy and Loss curve for Resnet50 CNN model

VGG-16 Model

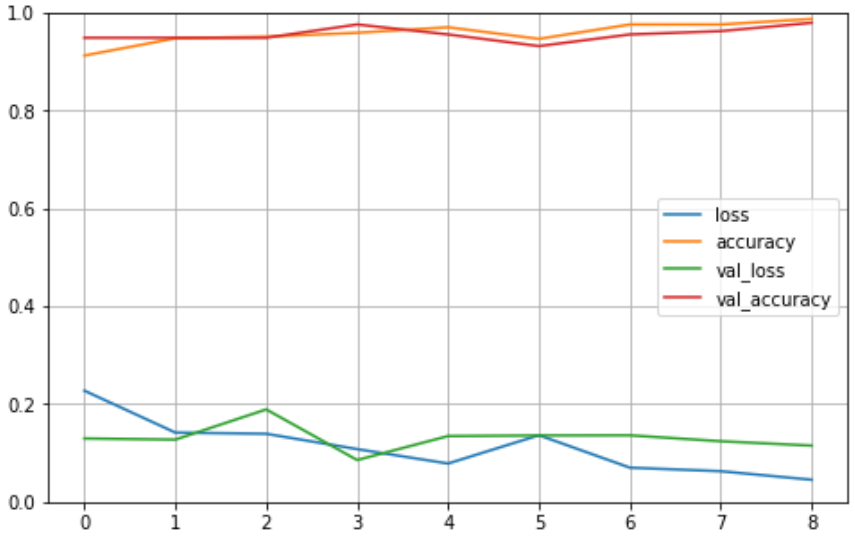


Figure 38: Accuracy and Loss curve for VGG-16 CNN model

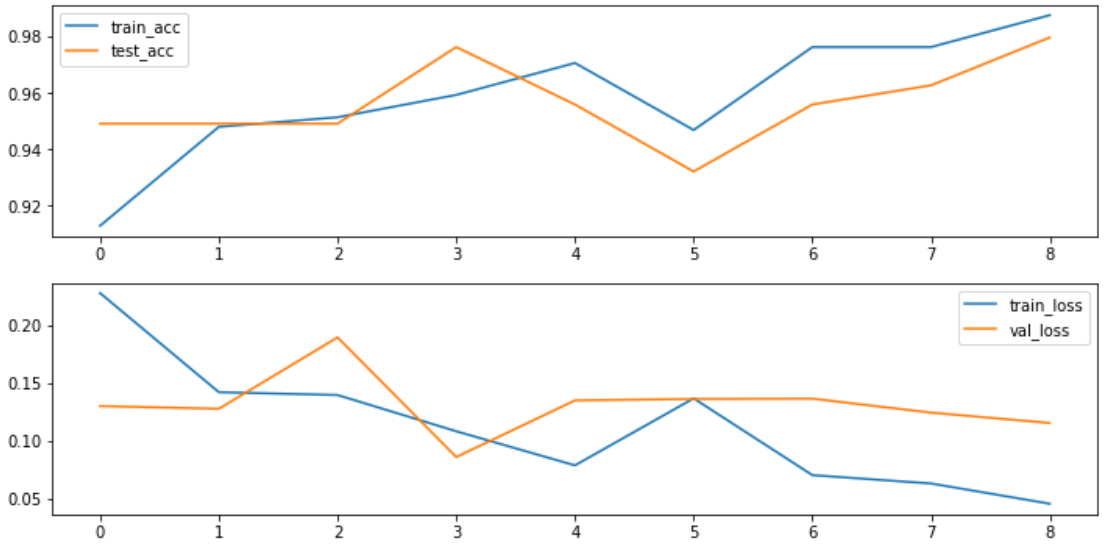
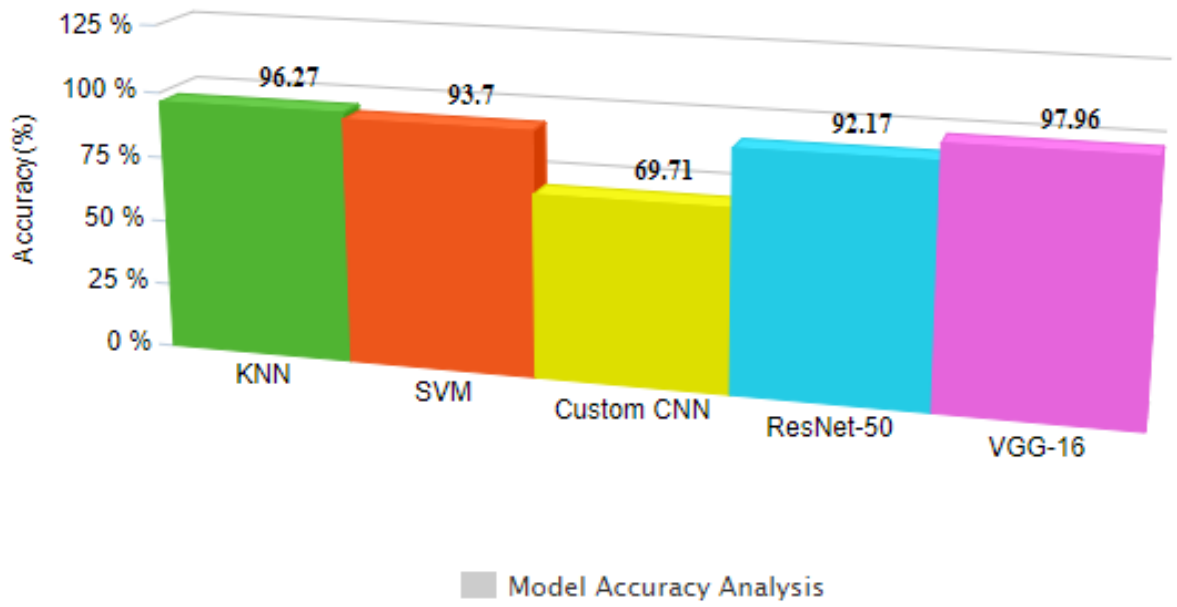


Figure 39: Training and Testing Accuracy & Loss for VGG-16 CNN model

The plot for accuracy metric comparing the performance of training accuracy and validation accuracy obtained during the training process is given in above figures. Both accuracy curves are steadily increasing according to the plot, with a faster ceiling level obtained for training accuracy. In addition, it shows a plot for the loss function comparing the behaviour of training loss and testing loss obtained during the training process. Both losses are systematically decreasing.

Model	Dataset	Accuracy Obtained
KNN	Wisconsin Dataset	96.27%
SVM	Wisconsin Dataset	93.7%
Custom CNN Model	Histopathological Image Dataset	69.71%
ResNest-15 Pre-Trained Model	Histopathological Image Dataset	92.17%
VGG-16 Pre-Trained Model	Histopathological Image Dataset	97.96%



All of these results clearly demonstrate VGG16's classification capabilities and its robust performance on our dataset.

4.2 Application of the Minor Project

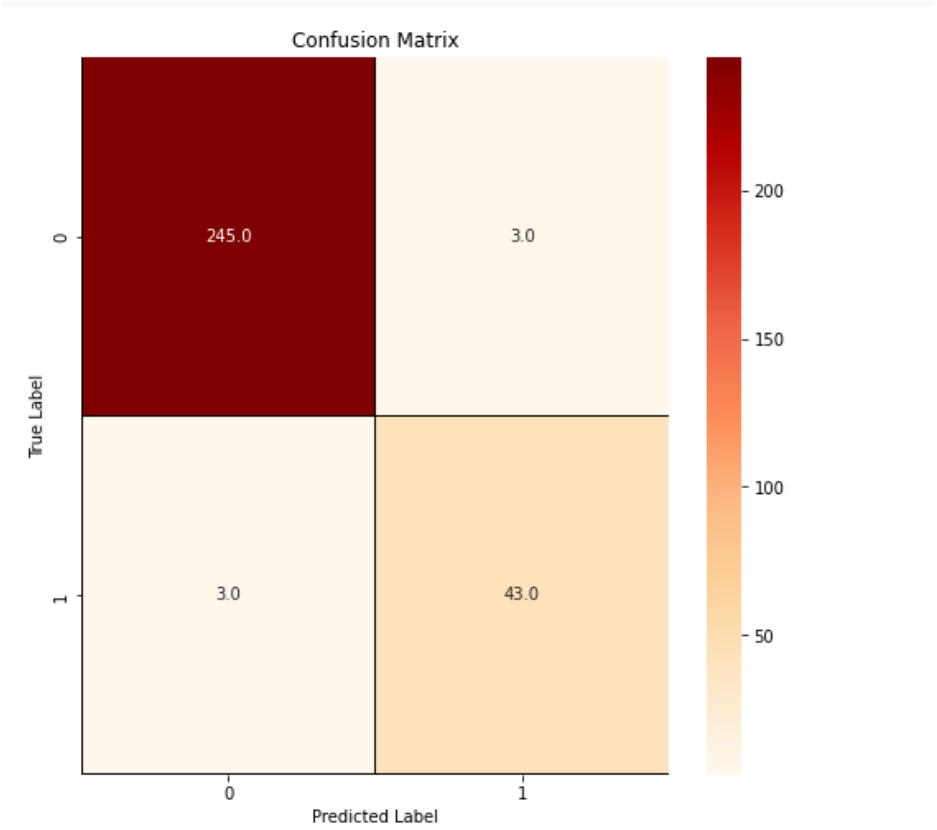
The application of computer-based learning models have emerged as a key area of cancer research. Several researchers have focused in recent years on developing systems, both hybrid and fully automatic systems, that could aid in the diagnosis, prognosis, and prediction of breast cancer outcomes by utilising Statistics and Artificial Intelligence. The development of these systems necessitates a variety of techniques, the most common of which are machine learning (ML) algorithms. Several scientific studies have published algorithms and nomograms for predicting the pathologic stage of patients with clinically localised cancer or Gleason score upgrading. Specifically, ML enables the integration or combination of various layers of data, such as those from medical images, laboratory results, clinical outcomes, biomarkers, and biological features, for improved patient prognosis and stratification toward personalised medicine. Deep learning approaches for extracting characteristics and improving the efficiency of medical image analysis have recently been developed. Deep learning is a type of machine learning that employs multilayer convolutional neural networks (CNN). Unlike other feature extraction techniques, CNN can extract image features directly from the dataset. Using convolution, this type of feature extraction extracts features from different parts of an image. Scientists have achieved promising results with CNNs for the diagnosis of breast cancer in recent years.

4.3 Limitation of the Minor Project

Our proposed model comparison approach is tested on a dataset with a small number of images. Despite the fact that the number of images is increased by using various data augmentation techniques, the performance of our proposed network could be further evaluated using a larger dataset. Furthermore, real-world data often differs from publicly available datasets.The main limitation of this project is that it relied on a secondary database such as Kaggle, and future studies should rely on primary data to improve the accuracy of the results related to breast cancer identification. If we could further experiment with real-world data, we could see how the model performs on it.According to confusion matrix,some images were misclassified,even after employing a variety of image processing techniques, the model still struggles to distinguish cancer cells from dense breast tissue.Despite the fact that data augmentation increases the number of images, due to a lack of new data, the model may not learn enough new features. However, the model performs admirably in the majority of the test images, correctly classifying the cases. Despite its minor issues, the model is robust.

There are a few small drawbacks to the project:

- It has a high computational complexity due to its several layers, and it also has a much reduced computational complexity due to operations such as maxpool.
- If the machine on which the model is being performed does not have a decent GPU, it will take a long time.
- One of the drawbacks of histopathological diagnostic procedures is that they are time-consuming and prone to sample errors.



4.4 Future Work

In future, we will be working on improving accuracies of the models and comparing them on the basis of other performance metrics. We will evaluate models on the basis of different optimizers and learning rates. Furthermore,we will be analysing other models like VGG-19 and DenseNet to bring out the best model for the detection of breast cancer. Once we come up with the most optimal model, we would try to develop an appropriate User Interface for the deployment of the model into a full-fledged Breast Cancer Detection System to be hosted on a server and be put into actual use .

References

[1] Masud, M., Eldin Rashed, A. E., & Hossain, M. S. (2020). Convolutional neural network-based models for diagnosis of breast cancer. Neural Computing and Applications. doi:10.1007/s00521-020-05394-5.

[2] Alanazi, S. A., Kamruzzaman, M. M., Islam Sarker, M. N., Alruwaili, M., Alhwaiti, Y., Alshammari, N., & Siddiqi, M. H. (2021). Boosting breast cancer detection using convolutional neural network. Journal of Healthcare Engineering, 2021.

[3] Ahmad, L. G., Eshlaghy, A. T., Poorebrahimi, A., Ebrahimi, M., & Razavi, A. R. (2013). Using three machine learning techniques for predicting breast cancer recurrence. J Health Med Inform, 4(124), 3.

[4] Huang, M. W., Chen, C. W., Lin, W. C., Ke, S. W., & Tsai, C. F. (2017). SVM and SVM ensembles in breast cancer prediction. PloS one, 12(1), e0161501.

[5] Agarap, A. F. M. (2018, February). On breast cancer detection: an application of machine learning algorithms on the wisconsin diagnostic dataset. In Proceedings of the 2nd international conference on machine learning and soft computing (pp. 5-9).

[6] Dey, S. CNN application on structured data-Automated Feature Extraction. URL: https://towardsdatascience.
com/cnnapplication-on-structured-data-automated-featureextraction-8f2cd28d9a7e.(accessed: 20.05. 2019).

[7] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

[8] Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. Insights into imaging, 9(4), 611-629.

[9] Gandhi, R. (2018). Support vector machine—introduction to machine learning algorithms. Towards Data Science, 7.

[10] Stolte, S., & Fang, R. (2020). A survey on medical image analysis in diabetic retinopathy. Medical image analysis, 64, 101742.

[11] Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., & Greenspan, H. (2018). GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. Neurocomputing, 321, 321-331.

[12] Dabiri, S., & Heaslip, K. (2018). Inferring transportation modes from GPS trajectories using a convolutional neural network. Transportation research part C: emerging technologies, 86, 360-371.

[13] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3), 211-252.

[14] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

[15] Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., ... & Webster, D. R. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. Jama, 316(22), 2402-2410.

[16] Bejnordi, B. E., Veta, M., Van Diest, P. J., Van Ginneken, B., Karssemeijer, N., Litjens, G., ... & CAMELYON16 Consortium. (2017). Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. Jama, 318(22), 2199-2210.

[17] Christ, P. F., Elshaer, M. E. A., Ettlinger, F., Tatavarty, S., Bickel, M., Bilic, P., ... & Menze, B. H. (2016, October). Automatic liver and lesion segmentation in CT using cascaded fully convolutional neural networks and 3D conditional random fields. In International conference on medical image computing and computer-assisted intervention (pp. 415-423). Springer, Cham.

[18] Hubel, D. H., & Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. The Journal of physiology, 195(1), 215-243.

[19] Assegie, T. A. (2021). An optimized K-Nearest Neighbor based breast cancer detection. Journal of Robotics and Control (JRC), 2(3), 115-118.

Additional Guidelines:-

1. There is no page limit as such but above mentioned titles and non-titles must be part of the project report. Ideally it should be of 25 to 50 pages.
2. Each report must be designed as per the format but the title of chapters, sub titles and number of chapters can be decided by the guide based on specific project.
3. Each project report must have a proper Table of Content.
4. Each report must have the page number starting from the chapter number 1.
5. Front page of the report should not be marked with any page number.
6. Declaration, Certificate and Table of Content must have numbers in roman (I, II, III….)
7. All references must be written in any standard style format, like IEEE/APA/etc.
8. All the references must be cited into the text at the appropriate place in the

report.

9. Similarity index must be less than 20%
10. Margins (1.5 on left, right, top, bottom), Line spacing 1.5 (as is present in this template), Body of the running text – 12 pts. Times new roman
11. Indentation of 1.27 on the first line of each paragraph.
12. Other formatting styles to be considered as is present in this template.